2nd AF

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:

Somashekar, et al.

Serial No. 09/733,232

Filed: December 7, 2000

For: TECHNIQUE FOR CONFIGURING NETWORK DELIVERABLE COMPONENTS

§ Group Art Unit: 2143
§
§ Examiner: Mauro Jr., Thomas J.
§
§ Atty. Dkt. No.: 5181-46501
§ P4338

CERTIFICATE OF MAILING
37 C.F.R. § 1.8

I hereby certify that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as First Class Mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date indicated below:

Robert C. Kowert
Name of Registered Representative

March 9, 2005
Date

Signature

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir/Madam:

Further to the Notice of Appeal mailed January 12, 2005, Appellants present this Appeal Brief. Appellants respectfully request that the Board of Patent Appeals and Interferences consider this appeal.

03/15/2005 MAHMED1 00000034 501505 09733232
01 FC:1402 500.00 DA

I. REAL PARTY IN INTEREST

As evidenced by the assignment recorded at Reel/Frame 011417/0149, the subject application is owned by Sun Microsystems, Inc., a corporation organized and existing under and by virtue of the laws of the State of Delaware, and now having its principal place of business at 4150 Network Circle, Santa Clara, CA 95054.

II. RELATED APPEALS AND INTERFERENCES

No other appeals, interferences or judicial proceedings are known which would be related to, directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1-54 are pending and rejected. The rejection of claims 1-54 is being appealed. A copy of claims 1-54 is included in the Claims Appendix hereto.

IV. STATUS OF AMENDMENTS

No amendments to the claims have been submitted subsequent to the final rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Networks, including the Internet, are useful mechanisms for enabling, programming, updating, and extending "smart" appliances and devices. For previously "dumb" devices like washers, air-conditioners, and sprinklers, powerful new capabilities are becoming possible using a combination of network access, "embedded" microchips, and "smart" software. Embedded systems still have limited local memory resources and only so much space is available for pre-installed services. But if services can be loaded on demand, then a small microprocessor can become a much more versatile computing

system. Where once a device could perform only one or two operations, now it can perform a wide variety of operations. This approach to embedded services simplifies management of the devices. The services can be maintained and administered in a centralized location, and can be delivered via the network as required. Users are no longer required to replace the entire device in order to upgrade to new services or capabilities. They simply load a new version of the controlling service.

Claim 1 is directed to a method for configuring pluggable components in a distributed computing environment or other network. Preference values for pluggable components on a first device are configured. For example, in some embodiments pluggable components may be referred to as bundles may be configured by a utility, such as a Bundle Configuration Utility (BCU), as described on page 5 of the specification. A BCU may enable the configuration of a set of properties associated with the pluggable components prior to distributing the pluggable components to other devices on the network. *See, e.g.*, FIGs. 3-7, page 12, line 28 – page 13, line 15; page 16, lines 7-17; page 18, line 20 – page 21, line 10.

After preference values for the pluggable components are modified or configured, the pluggable components are distributed to other devices via the network. For instance, as described on page 14 of the Specification, pluggable components may be distributed to one or more systems after being created, compiled, and configured, in one embodiment. There may be various methods of distributing pluggable components, such as via a command line interface or a batch file, or via a graphical user interface, according to various embodiments. *See, e.g.*, FIGs. 3 and 8-10; page 14, line 4- 18.

After distribution, the pluggable components are executable within the devices according to the configured preference values to provide service to users of the devices. For instance, pluggable components may be distributed to devices including embedded servers and the pluggable components may be accessed on the embedded servers to provide services on the embedded server devices. *See, e.g.*, page 5, lines 26 – 29; page 6, line 30 – page 7, line 3; page 10, lines 20-24; page 14, line 20-28.

Independent claim 21 is directed to a system in which pluggable components are configured on one device and distributed to other devices over a network. In some embodiments, the network may include the Internet. Configuring and distributing the pluggable components may be performed as described above regarding claim 1. As illustrated in FIG. 3, pluggable components, such as pluggable component 208A may be configured on configuration system 202 and distributed to various embedded server systems 212. *See, e.g.*, FIG. 3, 8, 9, and 10; page 10, line 27 – page 12, line 6; page 12, line 28 – page 14, line 28.

Independent claim 35 is directed to a device configured to configure preference values for pluggable components according to received user input. In some embodiments, a bundle configuration utility (BCU) may include various means of receiving user input for configuring pluggable components. For instance, user input specifying modifications to preference values for pluggable components may be received through a command line interface, a graphical user interface, or a batch file, according to different embodiments, and as described, for example, on page 13 of the Specification. The distributing of configured pluggable components is described above regarding claim 1. *See, e.g.*, FIGs. 3-10; page 10, line 27 – page 12, line 6; page 13, line 6 – page 14, line 2; page 18, line 20 – page 27, line 16.

Independent claim 45 is directed to a medium containing computer-executable program instructions implementing configuring and distributing of pluggable components. For example, a memory medium may store software programs for enabling a pluggable component configuration system, such as one including a BCU, to configure and distributing pluggable components. Examples of configuring and distributing of the pluggable components are described above regarding claim 1. *See, e.g.*, FIG. 3; page 12, lines 8-26.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 1-3, 8, 15, 17, 20-22, 29, 31, 34-36, 42, 44-46, 52 and 54 stand finally rejected under 35 U.S.C. § 102(e) as being anticipated by Young (U.S. Patent 6,560,606).

2. Claim 4 stands finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Young in view of Hammond (U.S. Patent 6,637,020).

3. Claims 5-6, 23, 37 and 47 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Young in view of Barrett et al. (U.S. Patent 6,611,876) (hereinafter "Barrett").

4. Claim 7 stands finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Young in view of Barrett in further view of Hammond.

5. Claims 9, 24, 38 and 48 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Young in view of Foltan et al. (U.S. Patent 6,667,972) (hereinafter "Foltan").

6. Claims 10-12, 25-26, 39-40 and 49-50 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Young in view of Semenzato (U.S. Patent 5,903,728).

7. Claims 13-14, 27-28, 41 and 51 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Young in view of Davis et al. (U.S. Patent 5,742,829) (hereinafter "Davis").

8. Claims 16, 18, 30 and 32 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Young in view of Lawrence (U.S. Patent 6,629,113).

9. Claims 19, 33, 43 and 53 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Young in view of Muschett et al. (U.S. Patent 6,026,437) (hereinafter "Muschett").

VII. ARGUMENT

First Ground of Rejection:

Claims 1-3, 8, 15, 17, 20-22, 29, 31, 34-36, 42, 44-46, 52 and 54 are rejected under 35 U.S.C. § 102(e) as being anticipated by Young (U.S. Patent 6,560,606). Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 1, 2, 3, 8 and 20:

Regarding claim 1, Young fails to teach a method comprising configuring preference values for one or more pluggable components on a first device; and distributing the one or more pluggable components to one or more other devices via a network subsequent to said configuring; wherein the one or more pluggable components are executable within the one or more other devices in accordance with the configured preference values to provide services to users of the one or more other devices.

In contrast, Young teaches a metering and processing system for processing metered information that incorporates configurable processing modules and a configuration manager. (Young, Abstract). Young's system includes "a mechanism for converting the metered information into session data, a processing unit for processing the session data, and a configuration manager." According to Young, the processing unit includes "an execution management framework, and a plurality of plug-ins for processing the session data as directed by the framework with each performing a sub-part of the calculations." According to Young, the plug-in modules reside on each processing unit and a configuration manager "generates a configuration file reflecting user selections of

configuration parameters for plug-in execution." (Young, column 3, lines 5-14). Young further teaches that the "configuration manager 150 generates a configuration file for each stage of the pipeline, preferably specifying the configuration data in XML format", that configuration files are "sent to each stage configuration module" and also that they are "sent to the execution management framework." (Young, column 10, lines 4 – 14).

Thus, Young's system **does not include** ***distributing*** plug-in modules to the individual machines ***subsequent to configuration*** of preference values for the plug-in modules on another device. Further, Young describes how "stage configuration module 416 receives configuration files 418 from the configuration manager 150, which define state operations as well as operation of the plug-ins." Thus, the configuration manager sends configuration information to each stage of the pipeline, but the plug-ins themselves already reside on each processing unit. The actual plug-in modules are not sent by the configuration manager, but rather reside on each individual processing unit of a distributed pipeline.

In response to the above arguments, the Examiner argues that Young's configuration files are pluggable components that are "executed within each stage to provide configuration and layout for the plug-ins and various other components" (See, Response to Arguments, section A). However, Young's configuration files are XML documents containing the preference values for his plug-ins. The configuration files, which the Examiner incorrectly equates to pluggable components, *are not executable within the other devices*. Specifically, Young teaches that the configuration files specify configuration data in XML format (Young, column 10, lines 4-7). As is well known in the art, XML is not an executable language or format. Instead, as Young describes, XML is "a standard, data structuring and formatting language" (Young, column 5, lines 19-20). Contrary to the Examiner's assertion, Young's configuration files are clearly not executable. No one of ordinary skill in the art would consider Young's configuration files to be executable plug-in modules.

The Examiner responds to the above arguments by contending in the Advisory Action that Young's configuration files are executed or *carried out* in order to configure the plug-ins for execution. However, the Examiner is interpreting the term "executed" incorrectly. The Examiner is interpreting "executed" as meaning something that is applied or carried out such as a soldier "executing an order". However, such an interpretation is not relevant to the art of configuring and distributing software, as one skilled in the art would understand it. In the art of configuring and distributing software in network environments, and in general in the art of software development, an executable software entity includes program instructions formatted or configured in a form to be loaded and executed as instructions to a processor or virtual machine. One skilled in the art would not consider applying or carrying out configuration preferences from a configuration file as executing a pluggable component. The Examiner argues that his interpretation of "executed" is within the "broadest reasonable interpretation." However, such an interpretation is clearly not "reasonable" within the art of configuring and distributing software in network environments, as understood by one skilled in the art. Claim 1 uses the term "executable", not "carried out." Neither Appellants' specification nor the general understanding in the art equates an executable pluggable component with a configuration file that is "carried out." In fact, Young itself distinguishes between configuration files and plug-ins.

Furthermore, even if Young's configuration files were executable, which Appellants maintain they are not, Young's configuration files would not be executable in accordance with the configured preference values *to provide services to users* of the other devices, as recited in claim 1. Young teaches that the configuration files contain configuration parameters for each stage configuration module and that they are used to configure the stages and plug-ins of Young's pipeline (Young, column 10, lines 15-56). Young does not describe his configuration files as providing services to users of the other devices. Instead, Young describes how other modules, such as the configuration manger, stage configuration manager, and execution management framework, access the configuration files in order to determine the proper functioning of Young's pipeline (Young, column 9, lines 16-33, and column 13, lines 17-29).

In response to the above remarks the Examiner asserts in the Advisory Action that Young's "metered data processing system provides services to users of other devices, specifically, to generate useful usage information regarding communication services usage to data consumers" (citing Young column 4, lines 53-57). However, the Examiner has not interpreted Young's entire metered data processing system as pluggable components, but instead equates (erroneously) that Young's configuration files are pluggable components. Thus, following the Examiner's interpretation and line of reasoning, to anticipate claim 1, Young's configuration files would need to be executable within the other devices to provide services to users of the one or more other devices. This is clearly not the case with Young's system. Young's configuration files do not provide any services to any users.

Additionally, following the Examiner's line of reasoning, in order to anticipate claim 1, Young would have to teach configuring preference values for his configuration files, which Young does not do. Claim 1 recites, in part, configuring preference values *for one or more pluggable components*. Thus, using the Examiner's interpretation of Young's configuration files as pluggable components, to anticipate claim 1, Young would have to teach configuring preference values for his configuration files. Instead, Young teaches using configuration files to store and communicate preferences for the stages and plug-in modules of his pipeline. Young also teaches that his configuration manager "generates a configuration file reflecting user selections of *configuration parameters for plug-in execution*" (emphasis added, Young, column 3, lines 12-14). Young clearly does not teach that his configuration manager configures parameters for his configuration files.

In response to the above argument, the Examiner, in the Advisory Action, asserts that "each pluggable component [in Young's system], i.e. configuration file, is comprised of preferences associated with specific parameters within the configuration file necessary to provide *configuration and layout to the stages and plug-ins*" (emphasis added). The Examiner's statement actually supports Appellant's arguments. As the Examiner states,

Young's configuration files include configuration and layout information for other processes, such as the individual stages and plug-ins. Thus, Young is not configuring preference values for his configuration file, but rather is using the configuration files to store preference values for other modules. Thus, the examiner's interpretation of Young's configuration files as pluggable modules is clearly incorrect. Although Young's configuration manager does send configuration information to each stage of the pipeline, the plug-ins themselves already reside on each processing unit. The actual plug-in modules are not sent by the configuration manager, but rather reside on each individual processing unit of a distributed pipeline. Thus, Young clearly does not anticipate claim 1.

Claim 15:

Regarding claim 15, contrary to the Examiner's assertion, Young does not teach that each of the pluggable components comprises a preferences file comprising the *preference values associated with the pluggable component.* In contrast, Young teaches the use of a single configuration file that is sent to every stage in a distributed pipeline, and that the configuration file contains the configuration information *for all plug-ins* executing on that device. According to Young, "[t]he configuration file is sent to each stage configuration module for *configuring the respective stage.*" (emphasis added, Young, column 10, lines 7-9). Young further teaches that a stage includes an input queue, an output queue, and a multithreading process space and that "[t]he process space processes a number of plug-ins ... under the control of an execution management framework." Furthermore, Young does not teach or suggest that a plug-in includes a preferences file that comprises its preference values. Thus, rather than teaching that *each pluggable component comprising a preference file* comprising the preference values associated with the pluggable component, Young teaches that a single configuration file includes the configuration information for a stage, which may include multiple plug-ins.

In response to the above argument, the Examiner maintains his interpretation of Young's configuration files as pluggable components. However, as discussed above for claim 1, such an interpretation of Young is clearly incorrect.

Additionally, the Examiner argues that each of Young's configuration files "is composed of a file containing preferences associated with specific parameters within the configuration file necessary to provide configuration and layout to the stages and plug-ins" (Final office action, Response to Arguments, section B). However, claim 15, recites that each of the one or more pluggable components comprises a preference file comprising the preference values *associated with the pluggable component.* Following the Examiner's interpretation of Young's configuration files as pluggable components, to anticipate claim 15, Young would have to teach that each of his configuration files comprise a preferences file including preference values associated with the configuration file. Young does not mention anything regarding preference files including preference values for his configuration files. Furthermore, the Examiner's own argument is that each configuration file provides configuration and layout for the stages and plug-ins, not for the configuration file itself.

The Examiner also argues, in the Advisory Action, that Young's configuration files hold information and parameters/values that direct the operation of the plug-ins and how the plug-ins behave. The Examiner further states, "[Young's] pluggable components are configuration/preference files and therefore teach the board language of the claim limitation." However, the Examiner is completely ignoring the specific limitation of claim 15 reciting that *each* of the pluggable components comprises a preference file comprising the *preference values associated with the pluggable component.* Young's configuration file, as admitted by the Examiner, comprises preference values for Young's (true) plug-ins – not the configuration file itself (that the Examiner is incorrectly relying on as a pluggable component).

Claim 17:

Regarding claim 17, Young fails to disclose wherein the one or more pluggable components are executable within the embedded server of each of the one or more other devices. The Examiner cites, Figures 3 and 5 and column 13, lines 23-49, where Young describes how his Execution management frameworks "collectively constitute an infrastructure that allows any type or number of plug-ins to be arranged and operated in any order *pursuant to a configuration file* associated with each session" (emphasis added). The Examiner is relying on Young's configuration file to correspond to the pluggable component of claim 17. However, Young, at the Examiner's cited passage clearly states that his configuration file is not arranged and operated by the execution management framework, but rather that the execution management framework organizes and executes other plug-ins according to the configuration files.

Furthermore, as noted above, Young's configuration files, contrary to the Examiner's contention, are not executable and thus cannot be executable on within an embedded server. Please see the discussion above regarding claim 1 for more details regarding why Young's configuration files are not executable.

Claim 21, 22 and 34:

Regarding claim 21, Young fails to teach a system comprising a first device configured to configure preference values for a plurality of pluggable components in accordance with user input; and distribute the plurality of pluggable components to a plurality of devices via a network subsequent to the configuring and in response to user input; and wherein the plurality of pluggable components are executable within the plurality of devices in accordance with the configured preference values to provide services to users of the plurality of devices. In contrast, as noted above regarding claim 1, Young teaches a metering and processing system for processing metered information that incorporates configurable processing modules and a configuration manager. (Young, Abstract). Please see the discussion above regarding claim 1 for more information on Young's system. According to Young, a configuration manager "generates a configuration file reflecting user selections of configuration

parameters for plug-in execution." (Young, column 3, lines 5-14). Young further teaches that the "configuration manager 150 generates a configuration file for each stage of the pipeline, preferably specifying the configuration data in XML format", that configuration files are "sent to each stage configuration module" and also that they are "sent to the execution management framework." (Young, column 10, lines 4 – 14). Thus, Young's system does not include *distributing* plug-in modules to the individual machines *subsequent to configuration* of preference values for the plug-in modules on another device. Instead, the configuration manager sends configuration information to each stage of the pipeline, but the plug-ins themselves already reside on each processing unit. The actual plug-in modules are not sent by the configuration manager, but rather reside on each individual processing unit of a distributed pipeline.

In response to the above arguments, the Examiner argues that Young's configuration files are pluggable components that are "executed within each stage to provide configuration and layout for the plug-ins and various other components" (See, Response to Arguments, section A). However, Young's configuration files are XML documents containing the preference values for his plug-ins. The configuration files, which the Examiner equates to pluggable components, *are not executable within the other devices*. For a more detailed argument regarding why Young's configuration files are not executable, please refer to the discussion of claim 1, above.

Furthermore, even if Young's configuration files were executable, which Applicants maintain they are not, they would not be executable in accordance with the configured preference values *to provide services to users* of the one or more other devices, as recited in claim 21. Young teaches that the configuration files contain configuration parameters for each stage configuration module and that they are used to configure the stages and plug-ins of Young's pipeline at three levels (Young, column 10, lines 15-56). Young does not describe the configuration as providing services to users of the other devices. Instead, Young describes how other modules, such as the configuration manger, stage configuration manager, and execution management

framework, access the configuration files in order to determine the proper functioning of Young's pipeline (Young, column 9, lines 16-33, and column 13, lines 17-29).

Additionally, following the Examiner's line of reasoning, in order to anticipate claim 21, Young would have to teach configuring preference values for his configuration files, which Young does not do. Claim 21 recites, in part, configuring preference values *for one or more pluggable components*. Thus, using the Examiner's interpretation of Young's configuration files as pluggable components, to anticipate claim 21, Young would have to teach configuring preference values for his configuration files. Young clearly does not do this. Instead, Young teaches using configuration files to store and communicate preferences for the stages and plug-in modules of his pipeline. Young also teaches that his configuration manager "generates a configuration file reflecting user selections of configuration parameters for plug-in execution" (Young, column 3, lines 12-14). Young does not teach that his configuration manager configures parameters for his configuration files.

Also, please see the arguments above regarding claim 1 that apply to claim 21 as well.

Claim 29:

Regarding claim 29, contrary to the Examiner's assertion, Young does not teach wherein each of the plurality of pluggable components comprises a preferences file comprising the preference values associated with the pluggable component. In contrast, Young teaches the use of a single configuration file that is sent to every stage in a distributed pipeline, and that the configuration file contains the configuration information *for all plug-ins* executing on that device. According to Young, "[t]he configuration file is sent to each stage configuration module for configuring the respective stage." (Young, column 10, lines 7-9). Young further teaches that a stage includes an input queue, an output queue, and a multithreading process space and that "[t]he process space processes a number of plug-ins ... under the control of an execution

management framework." Furthermore, Young does not teach or suggest that a plug-in includes a preferences file that comprises its preference values. Thus, rather than *each pluggable component comprising a preference file* comprising the preference values associated with the pluggable component, Young teaches that a single configuration file includes the configuration information for a stage, which may include multiple plug-ins.

Also, please see the arguments above regarding claim 15 as they also apply to claim 29 with equal force.

Claim 31:

Regarding claim 31, Young fails to disclose wherein the plurality of pluggable components are executable within the embedded servers of the plurality of devices. The Examiner cites, Figures 3 and 5 and column 13, lines 23-49, where Young describes how his Execution management frameworks "collectively constitute an infrastructure that allows any type or number of plug-ins to be arranged and operated in any order *pursuant to a configuration file* associated with each session" (emphasis added). Thus, Young, at the Examiner's cited passage clearly states that his configuration files (which the Examiner equates to the pluggable components of claim 31) are not arranged and operated by the execution management framework, but rather that the execution management framework organizes and executes other plug-ins according to the configuration files.

Furthermore, as noted above, Young's configuration files, contrary to the Examiner's contention, are not executable and thus cannot be executable on within an embedded server. Please see the discussion above regarding claim 1 for more details regarding why Young's configuration files are not executable.

Claim 35, 36 and 44:

Regarding claim 35, Young fails to teach a device configured to configure preference values for one or more pluggable components on the device in

accordance with received user input; and distribute the one or more pluggable components to one or more other devices via a network subsequent to the configuring and in response to user input; and wherein the one or more pluggable components are executable within the one or more other devices in accordance with the configured preference values to provide services to users of the one or more other devices. In contrast, as noted above regarding claims 1 and 21, Young teaches a metering and processing system for processing metered information that incorporates configurable processing modules and a configuration manager. (Young, Abstract). Please see the discussion above regarding claim 1 for more information on Young's system. According to Young, a configuration manager "generates a configuration file reflecting user selections of configuration parameters for plug-in execution." (Young, column 3, lines 5-14). Young further teaches that the "configuration manager 150 generates a configuration file for each stage of the pipeline, preferably specifying the configuration data in XML format", that configuration files are "sent to each stage configuration module" and also that they are "sent to the execution management framework." (Young, column 10, lines 4 – 14). Thus, Young's system does not include *distributing* plug-in modules to the individual machines *subsequent to configuration* of preference values for the plug-in modules on another device. Instead, the configuration manager sends configuration information to each stage of the pipeline, but the plug-ins themselves already reside on each processing unit. The actual plug-in modules are not sent by the configuration manager, but rather reside on each individual processing unit of a distributed pipeline.

In response to the above arguments, the Examiner argues that Young's configuration files are pluggable components that are "executed within each stage to provide configuration and layout for the plug-ins and various other components" (See, Response to Arguments, section A). However, Young's configuration files are XML documents containing the preference values for his plug-ins. The configuration files, which the Examiner equates to pluggable components, *are not executable within the other devices*. For a more detailed argument regarding why Young's configuration files are not executable, please refer to the discussion of claim 1, above.

Furthermore, even if Young's configuration files were executable, which Applicants maintain they are not, they would not be executable in accordance with the configured preference values *to provide services to users* of the one or more other devices, as recited in claim 35. Young teaches that the configuration files contain configuration parameters for each stage configuration module and that they are used to configure the stages and plug-ins of Young's pipeline at three levels (Young, column 10, lines 15-56). Young does not describe the configuration as providing services to users of the other devices. Instead, Young describes how other modules, such as the configuration manger, stage configuration manager, and execution management framework, access the configuration files in order to determine the proper functioning of Young's pipeline (Young, column 9, lines 16-33, and column 13, lines 17-29).

Additionally, following the Examiner's line of reasoning, in order to anticipate claim 35, Young would have to teach configuring preference values for his configuration files, which Young does not do. Claim 35 recites, in part, configuring preference values *for one or more pluggable components*. Thus, using the Examiner's interpretation of Young's configuration files as pluggable components, to anticipate claim 35, Young would have to teach configuring preference values for his configuration files. Young does not do this. Instead, Young teaches using configuration files to store and communicate preferences for the stages and plug-in modules of his pipeline. Young also teaches that his configuration manager "generates a configuration file reflecting user selections of configuration parameters for plug-in execution" (Young, column 3, lines 12-14). Young does not teach that his configuration manager configures parameters for his configuration files.

Also, please see the arguments above regarding claims 1 and 21 as they apply to claim 35 as well.

Claim 42:

Regarding claim 42, contrary to the Examiner's assertion, Young does not teach wherein each of the pluggable components comprises a preferences file comprising the *preference values associated with the pluggable component.* In contrast, Young teaches the use of a single configuration file that is sent to every stage in a distributed pipeline, and that the configuration file contains the configuration information *for all plug-ins* executing on that device. According to Young, "[t]he configuration file is sent to each stage configuration module for configuring the respective stage." (Young, column 10, lines 7-9). Young further teaches that a stage includes an input queue, an output queue, and a multithreading process space and that "[t]he process space processes a number of plug-ins ... under the control of an execution management framework." Furthermore, Young does not teach or suggest that a plug-in includes a preferences file that comprises its preference values. Thus, rather than *each pluggable component comprising a preference file* comprising the preference values associated with the pluggable component, Young teaches that a single configuration file includes the configuration information for a stage, which may include multiple plug-ins.

Also, please see the arguments above regarding claims 15 and 29 as they also apply to claim 42 with equal force.

Claim 45, 46 and 54:

Regarding claim 45, Young fails to teach a medium comprising program instructions that are computer-executable to implement configuring preference values for one or more pluggable components on a first device; and distributing the one or more pluggable components to one or more other devices via a network subsequent to said configuring; wherein the one or more pluggable components are executable within the one or more other devices in accordance with the configured preference values to provide services to users of the one or more other devices. In contrast, as noted above regarding claim 1, Young teaches a metering and processing system for processing metered information that incorporates configurable processing modules and a configuration manager. (Young, Abstract). Please see the discussion

above regarding claim 1 for more information on Young's system. According to Young, a configuration manager "generates a configuration file reflecting user selections of configuration parameters for plug-in execution." (Young, column 3, lines 5-14). Young further teaches that the "configuration manager 150 generates a configuration file for each stage of the pipeline, preferably specifying the configuration data in XML format", that configuration files are "sent to each stage configuration module" and also that they are "sent to the execution management framework." (Young, column 10, lines 4 – 14). Thus, Young's system does not include *distributing* plug-in modules to the individual machines *subsequent to configuration* of preference values for the plug-in modules on another device. Instead, the configuration manager sends configuration information to each stage of the pipeline, but the plug-ins themselves already reside on each processing unit. The actual plug-in modules are not sent by the configuration manager, but rather reside on each individual processing unit of a distributed pipeline.

In response to the above arguments, the Examiner argues that Young's configuration files are pluggable components that are "executed within each stage to provide configuration and layout for the plug-ins and various other components" (See, Response to Arguments, section A). However, Young's configuration files are XML documents containing the preference values for his plug-ins. The configuration files, which the Examiner equates to pluggable components, *are not executable within the other devices*. For a more detailed argument regarding why Young's configuration files are not executable, please refer to the discussion of claim 1, above.

Furthermore, even if Young's configuration files were executable, which Applicants maintain they are not, they would not be executable in accordance with the configured preference values *to provide services to users* of the one or more other devices, as recited in claim 45. Young teaches that the configuration files contain configuration parameters for each stage configuration module and that they are used to configure the stages and plug-ins of Young's pipeline at three levels (Young, column 10, lines 15-56). Young does not describe the configuration as providing services to users of the other devices. Instead, Young describes how other modules, such as the

configuration manger, stage configuration manager, and execution management framework, access the configuration files in order to determine the proper functioning of Young's pipeline (Young, column 9, lines 16-33, and column 13, lines 17-29).

Additionally, following the Examiner's line of reasoning, in order to anticipate claim 45, Young would have to teach configuring preference values for his configuration files, which Young does not do. Claim 45 recites, in part, configuring preference values *for one or more pluggable components*. Thus, using the Examiner's interpretation of Young's configuration files as pluggable components, to anticipate claim 45, Young would have to teach configuring preference values for his configuration files. Young does not do this. Instead, Young teaches using configuration files to store and communicate preferences for the stages and plug-in modules of his pipeline. Young also teaches that his configuration manager "generates a configuration file reflecting user selections of configuration parameters for plug-in execution" (Young, column 3, lines 12-14). Young does not teach that his configuration manager configures parameters for his configuration files.

Also, please see the arguments above regarding claims 1, 21 and 35 that apply to claim 45 as well.

Claim 52:

Regarding claim 52, Young fails to disclose wherein the one or more pluggable components are executable within the embedded server of each of the one or more other devices. The Examiner cites, Figures 3 and 5 and column 13, lines 23-49, where Young describes how his Execution management frameworks "collectively constitute an infrastructure that allows any type or number of plug-ins to be arranged and operated in any order *pursuant to a configuration file* associated with each session" (emphasis added). Thus, Young, at the Examiner's cited passage clearly states that his configuration files are not arranged and operated by the execution management

framework, but rather that the execution management framework organizes and executes other plug-ins according to the configuration files.

Furthermore, as noted above, Young's configuration files, contrary to the Examiner's contention, are not executable and thus cannot be executable on within an embedded server. Please see the discussion above regarding claim 1 for more details regarding why Young's configuration files are not executable.

Second Ground of Rejection:

Claim 4 is rejected under 35 U.S.C. § 103(a) as being unpatentable over Young in view of Hammond (U.S. Patent 6,637,020). Appellants traverse this rejection for at least the following reasons.

Claim 4:

Claim 4 is patentable for at least the reasons presented above regarding its independent claim (claim 1).

Third Ground of Rejection:

Claims 5-6, 23, 37 and 47 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Young in view of Barrett et al. (U.S. Patent 6,611,876) (hereinafter "Barrett"). Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 5 and 6:

Claims 5 and 6 are patentable for at least the reasons presented above regarding their independent claim (claim 1).

Claim 23:

Claim 23 is patentable for at least the reasons presented above regarding its independent claim (claim 21).

Claim 37:

Claim 37 is patentable for at least the reasons presented above regarding its independent claim (claim 35).

Claim 47:

Claim 47 is patentable for at least the reasons presented above regarding its independent claim (claim 45).

Fourth Ground of Rejection:

Claim 7 is rejected under 35 U.S.C. § 103(a) as being unpatentable over Young in view of Barrett in further view of Hammond. Appellants traverse this rejection for at least the following reasons.

Claim 7:

Claim 7 is patentable for at least the reasons presented above regarding its independent claim (claim 1).

Fifth Ground of Rejection:

Claims 9, 24, 38 and 48 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Young in view of Foltan et al. (U.S. Patent 6,667,972) (hereinafter "Foltan"). Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

Claim 9:

Claim 9 is patentable for at least the reasons presented above regarding its independent claim (claim 1).

Claim 24:

Claim 24 is patentable for at least the reasons presented above regarding its independent claim (claim 21).

Claim 38:

Claim 38 is patentable for at least the reasons presented above regarding its independent claim (claim 35).

Claim 48:

Claim 48 is patentable for at least the reasons presented above regarding its independent claim (claim 45).

Sixth Ground of Rejection:

Claims 10-12, 25-26, 39-40 and 49-50 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Young in view of Semenzato (U.S. Patent 5,903,728). Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 10 and 11:

Regarding claim 10, Young in view of Semenzato fails to teach wherein the one or more pluggable components is a plurality of pluggable components, wherein each of the plurality of pluggable components are copies of a first pluggable component. The Examiner admits that Young fails to teach wherein each of the

pluggable components is a copy of the first pluggable component and relies upon Semenzato for this teaching.

The Examiner cites column 3, lines 46-54 and column 6, lines 18-42 of Semenzato. However, at the cited portions Semenzato teaches the use of the Unix fork() system call to duplicate an executing process before executing a plug-in module. The Examiner argues that the Unix fork() call taught by Semenzato can be used to duplicate the configuration files of Young. However, as described above, the configuration files of Young are XML documents, not executable files. The fork() command is a very specialized function call to duplicate a Unix process. The fork() system call simply cannot be used to duplicate the configuration files of Young.

Additionally, even if the configuration files of Young were executable, which Appellants contend they are not, it is unclear how Semenzato's use of the fork() system call could be combined with Young's pipeline configuration system. For example, Young teaches that the configuration files are distributed to each pipeline server via HTTP (Young, column 10, lines 11-14). Once a configuration file is on a pipeline server, there is no need to duplicate it using the fork() system call (or any other mechanism), even if the configuration files were executable, which they are not. Furthermore, it makes no sense to use the fork() system call to duplicate a process before distribution, as fork() only duplicates an executing process in system memory and such a duplicate (of a running process) is not appropriate for distribution via HTTP, as taught by Young.

In response to the above remarks, the Examiner, in the Advisory Action, merely restates his contention that Semenzato teaches duplicating a plug-in controller and cites the same portions of Semenzato discussed above. However, the Examiner fails to provide any argument or explanation regarding how Semenzato's use of the Unix fork() command can be utilized to duplicate Young's XML-based configuration files. Nor does the Examiner provide any rebuttal regarding Appellants' argument that processes duplicated using the fork() command cannot be distributed using HTTP as taught by Young.

The Examiner's interpretation of Young's configuration files as pluggable modules, is clearly not compatible with duplication using the fork() system call of Semenzato. Thus, any combination of Young and Semenzato would not suggest a system in which each of a plurality of pluggable components are copies of a first pluggable component.

Claim 12:

Regarding claim 12, Young in view of Semenzato fails to teach sending each of the plurality of pluggable components to the corresponding one of the plurality of devices via the network. The Examiner cites portions of Young that describe distributing configuration files via HTTP (Young, fig 3, column 8, lines 48-58, column 10, lines 9-20, column 13, lines 32-42 and column 17, lines 16-21). However, claim 12 also requires that each of the plurality of pluggable components *be a copy of a first pluggable component* (via claim 10). Semenzato teaches the use of a fork() system call to duplicate executing processes and indeed the Examiner relies upon this teaching in the rejection of claim 10. However, as described above regarding claim 10, the fork() system call is incompatible with the XML based configuration files of Young. Furthermore, since the fork() system call duplicates processes executing in system memory, the resulting duplicates cannot be distributed via HTTP as suggested by the Examiner. Thus, the combination of Young and Semenzato fails to teach sending each of the plurality of pluggable components to the corresponding one of the plurality of devices via the network.

Claims 25:

Regarding claim 25, Young in view of Semenzato fails to teach wherein each of the plurality of pluggable components are copies of a first pluggable component. The Examiner admits that Young fails to teach wherein each of the pluggable components is a copy of the first pluggable component and relies upon Semenzato for this teaching.

The Examiner cites column 3, lines 46-54 and column 6, lines 18-42 of Semenzato. However, at the cited portions Semenzato teaches the use of the Unix fork() system call to duplicate a platform process before executing a plug-in module. The Examiner argues that the Unix fork() call taught by Semenzato can be used to duplicate the configuration files of Young. However, as described above regarding the § 102 rejection of claim 1, the configuration files of Young are XML documents, not executable files. The fork() system call cannot be used to duplicate the configuration files of Young. Additionally, even if the configuration files of Young were executable, which Appellants contend they are not, it is unclear how Semenzato's use of the fork() system call could be combined with Young's pipeline configuration system. For example, Young teaches that the configuration files are distributed to each pipeline server via HTTP (Young, column 10, lines 11-14). Once a configuration file is on a pipeline server, there is no need to then duplicate it using the fork() system call (if the configuration files were executable, which they are not). Additionally, it makes no sense to use the fork() system call to duplicate a process before distribution, as fork() only duplicates an executing process in system memory and such a duplicate is not appropriate for distribution via HTTP as taught by Young.

Also, please see the above arguments regarding claim 10, as they apply to claim 25 as well.

Claim 26:

Regarding claim 26, Young in view of Semenzato fails to teach sending each of the plurality of pluggable components to the corresponding one of the plurality of devices via the network. The Examiner cites portions of Young that describe distributing configuration files via HTTP (Young, fig 3, column 8, lines 48-58, column 10, lines 9-20, column 13, lines 32-42 and column 17, lines 16-21). However, claim 26 also requires that each of the plurality of pluggable components be a copy of a first pluggable component (via claim 25). Semenzato teaches the use of a fork() system call to

duplicate executing processes and indeed the Examiner relies upon this teaching in the rejection of claim 10. However, as described above regarding claim 10, the fork() system call is incompatible with the XML based configuration files of Young. Furthermore, since the fork() system call duplicates processes executing in system memory, the resulting duplicates cannot be distributed via HTTP as suggested by the Examiner. Thus, the combination of Young and Semenzato fails to teach sending each of the plurality of pluggable components to the corresponding one of the plurality of devices via the network.

Claim 39:

Regarding claim 39, Young in view of Semenzato fails to teach wherein the one or more pluggable components is a plurality of pluggable components, wherein each of the plurality of pluggable components are copies of a first pluggable component. The Examiner admits that Young fails to teach wherein each of the pluggable components is a copy of the first pluggable component and relies upon Semenzato for this teaching.

The Examiner cites column 3, lines 46-54 and column 6, lines 18-42 of Semenzato. However, at the cited portions Semenzato teaches the use of the Unix fork() system call to duplicate a platform process before executing a plug-in module. The Examiner argues that the Unix fork() call taught by Semenzato can be used to duplicate the configuration files of Young. However, as described above regarding the § 102 rejection of claim 1, the configuration files of Young are XML documents, not executable files. The fork() system call cannot be used to duplicate the configuration files of Young. Additionally, even if the configuration files of Young were executable, which Appellants contend they are not, it is unclear how Semenzato's use of the fork() system call could be combined with Young's pipeline configuration system. For example, Young teaches that the configuration files are distributed to each pipeline server via HTTP (Young, column 10, lines 11-14). Once a configuration file is on a pipeline server, there is no need to then duplicate it using the fork() system call (if the

configuration files were executable, which they are not). Additionally, it makes no sense to use the fork() system call to duplicate a process before distribution, as fork() only duplicates an executing process in system memory and such a duplicate is not appropriate for distribution via HTTP as taught by Young.

Also, please see the above arguments regarding claims 10 and 25, as they apply to claim 39 as well.

Claim 40:

Regarding claim 40, Young in view of Semenzato fails to teach wherein, in said distributing, the processor is further operable to send each of the plurality of pluggable components to the corresponding one of the plurality of devices via the network. The Examiner cites portions of Young that describe distributing configuration files via HTTP (Young, fig 3, column 8, lines 48-58, column 10, lines 9-20, column 13, lines 32-42 and column 17, lines 16-21). However, claim 26 also requires that each of the plurality of pluggable components be a copy of a first pluggable component (via claim 25). Semenzato teaches the use of a fork() system call to duplicate executing processes and indeed the Examiner relies upon this teaching in the rejection of claim 10. However, as described above regarding claim 10, the fork() system call is incompatible with the XML based configuration files of Young. Furthermore, since the fork() system call duplicates processes executing in system memory, the resulting duplicates cannot be distributed via HTTP as suggested by the Examiner. Thus, the combination of Young and Semenzato fails to teach sending each of the plurality of pluggable components to the corresponding one of the plurality of devices via the network.

Claim 49:

Regarding claim 49, Young in view of Semenzato fails to teach wherein the one or more pluggable components is a plurality of pluggable components, wherein each of the plurality of pluggable components are copies of a first pluggable component. The Examiner admits that Young fails to teach wherein each of the

pluggable components is a copy of the first pluggable component and relies upon Semenzato for this teaching.

The Examiner cites column 3, lines 46-54 and column 6, lines 18-42 of Semenzato. However, at the cited portions Semenzato teaches the use of the Unix fork() system call to duplicate a platform process before executing a plug-in module. The Examiner argues that the Unix fork() call taught by Semenzato can be used to duplicate the configuration files of Young. However, as described above regarding the § 102 rejection of claim 1, the configuration files of Young are XML documents, not executable files. The fork() system call cannot be used to duplicate the configuration files of Young. Additionally, even if the configuration files of Young were executable, which Appellants contend they are not, it is unclear how Semenzato's use of the fork() system call could be combined with Young's pipeline configuration system. For example, Young teaches that the configuration files are distributed to each pipeline server via HTTP (Young, column 10, lines 11-14). Once a configuration file is on a pipeline server, there is no need to then duplicate it using the fork() system call (if the configuration files were executable, which they are not). Additionally, it makes no sense to use the fork() system call to duplicate a process before distribution, as fork() only duplicates an executing process in system memory and such a duplicate is not appropriate for distribution via HTTP as taught by Young.

Also, please see the above arguments regarding claim 10, as they apply to claim 49 as well.

Claim 50:

Regarding claim 12, Young in view of Semenzato fails to teach wherein, in said distributing, the program instructions are further computer-executable to implement sending each of the plurality of pluggable components to the corresponding one of the plurality of devices via the network. The Examiner cites portions of Young that describe distributing configuration files via HTTP (Young, fig 3,

column 8, lines 48-58, column 10, lines 9-20, column 13, lines 32-42 and column 17, lines 16-21). However, claim 26 also requires that each of the plurality of pluggable components be a copy of a first pluggable component (via claim 25). Semenzato teaches the use of a fork() system call to duplicate executing processes and indeed the Examiner relies upon this teaching in the rejection of claim 10. However, as described above regarding claim 10, the fork() system call is incompatible with the XML based configuration files of Young. Furthermore, since the fork() system call duplicates processes executing in system memory, the resulting duplicates cannot be distributed via HTTP as suggested by the Examiner. Thus, the combination of Young and Semenzato fails to teach sending each of the plurality of pluggable components to the corresponding one of the plurality of devices via the network.

Seventh Ground of Rejection:

Claims 13-14, 27-28, 41 and 51 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Young in view of Davis et al. (U.S. Patent 5,742,829) (hereinafter "Davis"). Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

Claim 13:

In regards to claim 13, Young in view of Davis fails to teach or suggest executing the batch file on the first device. The Examiner cites portions of Davis (column 11, lines 50-67, column 12, lines 61 - column 13, line 38) that describe using a SMSLS batch file to configure a client computer from a server computer. However, Davis teaches that "although the SMSLS batch file ... [is] located on the client server, the processing (or execution) of these files *occurs on the processor of the client computer*" (emphasis added, Davis, column 11, lines 60-64). Thus, **Davis teaches away** from executing the batch file on the first device, which corresponds to the server machine in Davis' system. When modified in view of the cited teachings of Davis, Young's system would include executing a batch file on each of Young's pipeline servers. Therefore, the

Examiner's proposed combination of Young and Davis fails to teach or suggest executing the batch file on the first device.

In response to the above remarks, the Examiner, in the Advisory action, contends that Appellants are attempting to argue each cited reference piecemeal. However, the Examiner has completely ignored the portion of Appellants' argument regarding the combination of Young and Davis. Specifically, as noted above, Appellants' argue that the Examiner's combination of Young in view of Davis would not result in a system that includes all the limitations of claim 13. Specifically, the Young in view of Davis would execute a batch file on each of Young's pipeline servers, rather than on the first device, as recited by claim 13.

The Examiner further argues that Davis is relied upon only to teach that a batch file can be executed. However, as admitted by the Examiner, Young fails to teach the use of a batch file at all, thus, not only does the Examiner need to provide motivation to modify Young to include executing a batch file, but also must provide motivation for modifying Young to include executing a batch file on Young's management framework server. The Examiner has clearly not done so. Thus, Young and Davis, both singly and in the Examiner's proposed combination, fail to teach or suggest executing a batch file on a first device, as recited in claim 13.

Furthermore, the combination of Young and Davis also fails to teach or suggest generating a batch file comprising one or more configuration entries for the one or more pluggable components, wherein each configuration entry includes information specifying one of the one or more pluggable components; information specifying one of the preference values for the specified pluggable component; and a new value for the specified preference values, as recited in claim 13. In contrast, Davis teaches that his batch file invokes a client setup program, executed on a client computer, that copies various files of a client operating system onto the client computer. Davis does not mention including in his batch file configuration entries for pluggable components. Davis also fails to teach that such a configuration entry includes

information specifying a pluggable component, information specifying a preference values for the specified pluggable component, or a new value for the specified preference value. As Young fails to teach any use of a batch file, as admitted by the Examiner, Young fails to correct any of the deficiencies of Davis' teachings. Thus, the Examiner's proposed combination of Young in view of Davis fails to result in a system that teaches the limitations of claim 13 as described above. Instead, such a combination of Young and Davis would result in a system in which batch files are executed on the individual machines of Young's system to invoke individual, custom setup programs.

Claim 14:

Regarding claim 14, Young in view of Davis does not teach or suggest wherein the batch file further comprises one or more distribution entries, wherein each distribution entry includes: information specifying one of the one or more pluggable components; and information specifying one of the one or more other devices as a destination device for the specified pluggable component; wherein said executing the batch file comprises executing each of the one or more distribution entries in the batch file; and wherein said executing each of the one or more distribution entries comprises sending the specified pluggable component to the specified destination device via the network. The Examiner relies upon Davis to teach executing a batch file and cites column 11, lines 50-67 and column 12, line 61 – column 13, line 38, where Davis describes installing software on heterogeneous computer systems by executing a custom batch file on client computers.

The Examiner argues only that Davis' system "executes batch file that executes a series of commands, specified in the file, to configure heterogeneous computer systems." However, Davis fails to teach including in a batch file distribution entries including information specifying a destination device for a pluggable component and further fails to teach wherein executing such distribution entries in a batch file comprises sending a specified pluggable component to the specified destination device.

In contrast, Davis teaches executing a batch file on the client machine (i.e. destination device) that invokes a custom client setup executable that in turn "copies the files representing the programs in the program list of the domain initialization file onto the client" (Davis, column 13, lines 1-3; column 14, lines 11-13). Thus, Davis teaches the use of a custom setup program (or executable) to copy files from the server to the client device. A setup program copying files from a server to a client not only fails to teach or suggest using distribution entries in a batch file to copy files, but actually teaches away from using distribution entries in a batch file to copy files.

Furthermore, the Examiner's proposed combination of Young in view of Davis does not suggest a system that includes distribution entries in a batch file specifying pluggable components and destination devices for the pluggable components, wherein executing the distribution entries comprises sending a specified pluggable component to a specified destination device. Instead, the combination of Young and Davis would only result in a system that uses a batch file to invoke setup programs on each of Young's distributed machines that in turn copy files from Young's server to the respective distributed machines.

Claim 27:

In regards to claim 27, Young in view of Davis fails to teach or suggest a first device configured to execute the batch file. The Examiner cites portions of Davis (column 11, lines 50-67, column 12, lines 61 - column 13, line 38) that describe using a SMSLS batch file to configure a client computer from a server computer. However, Davis teaches that "although the SMSLS batch file ... [is] located on the client server, the processing (or execution) of these files occurs on the processor of the client computer" (Davis, column 11, lines 60-64). Thus, Davis actually teaches away from executing the batch file on the first device, which corresponds to the server machine in Davis' system. When modified in view of the cited teaches of Davis, Young's system would include executing a batch file on each of Young's pipeline servers. Therefore, the Examiner's

proposed combination of Young and Davis files to teach a first device executing the batch file.

Furthermore, the combination of Young and Davis further fails to teach or suggests a first device configured to generate a batch file comprising one or more configuration entries for the one or more pluggable components, wherein each configuration entry includes information specifying one of the one or more pluggable components; information specifying one of the preference values for the specified pluggable component; and a new value for the specified preference values, as recited in claim 27. In contrast, Davis teaches that his batch file invokes a client setup executable that copies various files of the client operating system onto the client computer. Davis does not mention including, in his batch file, configuration entries for pluggable components. Davis also fails to teach that such a configuration entry includes information specifying a pluggable component, information specifying a preference values for the specified pluggable component, and a new value for the specified preference value. As Young fails to teach the any use of a batch file, as admitted by the Examiner, Young fails to correct any of the deficiencies of Davis' teachings.

Also, please refer to the arguments above regarding claim 13, as they apply to claim 27 as well.

Claim 28:

Regarding claim 28, Young in view of Davis does not teach or suggest wherein the batch file further comprises one or more distribution entries, wherein each distribution entry includes: information specifying one of the one or more pluggable components; and information specifying one of the one or more other devices as a destination device for the specified pluggable component; wherein said executing the batch file comprises executing each of the one or more distribution entries in the batch file; and wherein said executing each of the one or more distribution entries comprises sending the specified pluggable component to the

specified destination device via the network. The Examiner relies upon Davis to teach executing a batch file and cites column 11, lines 50-67 and column 12, line 61 – column 13, line 38, where Davis describes installing an software on heterogeneous computer systems by executing a custom batch file on client computers.

The Examiner argues that Davis' system "executes batch file that executes a series of commands, specified in the file, to configure heterogeneous computer system." However, Davis fails to teach including in a batch file distribution entries including information specifying a destination device for a pluggable component and further fails to teach (or suggest) wherein executing such distribution entries in a batch file comprises sending a specified pluggable component to the specified destination device. Instead, Davis teaches executing a batch file on the client machine (i.e. destination device) that invokes a custom client setup executable that in turn "copies the files representing the programs in the program list of the domain initialization file onto the client" (Davis, column 13, lines 1-3; column 14, lines 11-13). Thus, Davis teaches the use of a custom setup program (or executable) to copy files from the server to the client device. A setup program copying files from a server to a client not only fails to teach or suggest using distribution entries in a batch file to copy files, but actually teaches away from using distribution entries in a batch file to copy files.

Thus, the Examiner's proposed combination of Young in view of Davis does not suggest a system that includes distribution entries in a batch file specifying pluggable components and destination devices for the pluggable components, wherein executing the distribution entries comprises sending a specified pluggable component to a specified destination device. Instead, the combination of Young and Davis would only result in a system that may use a batch file to invoke setup programs that in turn copy files from a server to a client machine, as taught by Davis.

Claim 41:

In regard to claim 41, Young in view of Davis fails to teach executing the batch file on the first device. The Examiner cites portions of Davis (column 11, lines 50-67, column 12, lines 61 - column 13, line 38) that describe using a SMSLS batch file to configure a client computer from a server computer. However, Davis teaches that "although the SMSLS batch file ... [is] located on the client server, the processing (or execution) of these files occurs on the processor of the client computer" (Davis, column 11, lines 60-64). Thus, Davis actually teaches away from executing the batch file on the first device, which corresponds to the server machine in Davis' system. When modified in view of the cited teaches of Davis, Young's system would include executing a batch file on each of Young's pipeline servers. Therefore, the Examiner's proposed combination of Young and Davis files to teach executing the batch file on the first device.

Furthermore, the combination of Young and Davis further fails to teach generating a batch file comprising one or more configuration entries for the one or more pluggable components, wherein each configuration entry includes information specifying one of the one or more pluggable components; information specifying one of the preference values for the specified pluggable component; and a new value for the specified preference values, as recited in claim 41. In contrast, Davis teaches that his batch file invokes a client setup executable that copies various files of the client operating system onto the client computer. Davis does not mention including, in his batch file, configuration entries for pluggable components. Davis also fails to teach that such a configuration entry includes information specifying a pluggable component, information specifying a preference values for the specified pluggable component, and a new value for the specified preference value. As Young fails to teach the any use of a batch file, as admitted by the Examiner, Young fails to correct any of the deficiencies of Davis' teachings.

Also, please refer to the arguments above regarding claim 13, as they apply to claim 41 as well.

Claim 51:

In regard to claim 51, Young in view of Davis fails to teach generating a batch file comprising one or more configuration entries for the one or more pluggable components, wherein each configuration entry includes information specifying one of the one or more pluggable components; information specifying one of the preference values for the specified pluggable component; and a new value for the specified preference values, as recited in claim 41. In contrast, Davis teaches that his batch file invokes a client setup executable that copies various files of the client operating system onto the client computer. Davis does not mention including, in his batch file, configuration entries for pluggable components. Davis also fails to teach that such a configuration entry includes information specifying a pluggable component, information specifying a preference values for the specified pluggable component, and a new value for the specified preference value. As Young fails to teach the any use of a batch file, as admitted by the Examiner, Young fails to correct any of the deficiencies of Davis' teachings.

Furthermore, the combination of Young and Davis further fails to teach executing each of the one or more configuration entries in the batch file. The Examiner cites portions of Davis (column 11, lines 50-67, column 12, lines 61 - column 13, line 38) that describe using a SMSLS batch file to configure a client computer from a server computer. However, Davis teaches that "although the SMSLS batch file ... [is] located on the client server, the processing (or execution) of these files occurs on the processor of the client computer" (Davis, column 11, lines 60-64). Thus, Davis actually teaches away from executing the batch file on the first device, which corresponds to the server machine in Davis' system. When modified in view of the cited teaches of Davis, Young's system would include executing a batch file on each of Young's pipeline servers. Therefore, the Examiner's proposed combination of Young and Davis files to teach executing the batch file on the first device.

Also, please refer to the arguments above regarding claim 13, as they apply to claim 51 as well.

Eighth Ground of Rejection:

Claims 16, 18, 30 and 32 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Young in view of Lawrence (U.S. Patent 6,629,113). Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

Claim 16:

Claim 16 is patentable for at least the reasons presented above regarding its independent claim (claim 1).

Claim 18:

Regarding claim 18, Young in view of Lawrence fails to teach or suggest wherein the embedded servers include Java Embedded Servers. The Examiner argues that since JAVA was well known at the time Appellants' invention was made, and since Young teaches that object oriented technologies may be used to implement Young's plug-ins and execution frameworks, it would be obvious "to use JAVA as the OOP language for the plug-ins and execution framework in order to capture the portability it provides and the powerful concepts of encapsulation, polymorphism, and inheritance." However, the use of JAVA as a programming language to generate Young's plug-ins and execution frameworks does not teach, suggest, or imply using JAVA embedded servers. The Examiner has failed to provide any suggestion or motivation to use JAVA embedded servers in Young's system. Instead, the Examiner has only argued the use of JAVA as a programming language for implementing the plug-ins and execution frameworks of Young's system. Java-based programs and applications have traditionally been executed on machines under environments that do not include JAVA embedded servers. Not all software created using the JAVA object oriented programming language is executed on JAVA embedded servers. The Examiner has not provided any reasons why it would be obvious to include JAVA embedded servers in Young's system. Young and Lawrence,

either single or in combination, fail to mention, teach or suggest anything regarding JAVA embedded servers.

Claim 30:

Claim 30 is patentable for at least the reasons presented above regarding its independent claim (claim 21).

Claim 32:

Regarding claim 32, Young in view of Lawrence fails to teach or suggest wherein the embedded servers include Java Embedded Servers. The Examiner argues that since JAVA was well known at the time Appellants' invention was made, and since Young teaches that object oriented technologies may be used to implement Young's plug-ins and execution frameworks, it would be obvious "to use JAVA as the OOP language for the plug-ins and execution framework in order to capture the portability it provides and the powerful concepts of encapsulation, polymorphism, and inheritance." However, the use of JAVA as a programming language to generate Young's plug-ins and execution frameworks does not teach, suggest, or imply using JAVA embedded servers. The Examiner has failed to provide any suggestion or motivation to use JAVA embedded servers in Young's system. Instead, the Examiner has only argued the use of JAVA as a programming language for implementing the plug-ins and execution frameworks of Young's system. Java-based programs and applications have traditionally been executed on machines under environments that do not include JAVA embedded servers. Not all software created using the JAVA object oriented programming language is executed on JAVA embedded servers. The Examiner has not provided any reasons why it would be obvious to include JAVA embedded servers in Young's system. Young and Lawrence, either single or in combination, fail to mention, teach or suggest anything regarding JAVA embedded servers.

Ninth Ground of Rejection:

Claims 19, 33, 43 and 53 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Young in view of Muschett et al. (U.S. Patent 6,026,437) (hereinafter "Muschett"). Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

Claim 19:

Claim 19 is patentable for at least the reasons presented above regarding its independent claim (claim 1).

Claim 33:

Claim 33 is patentable for at least the reasons presented above regarding its independent claim (claim 21).

Claim 43:

Claim 43 is patentable for at least the reasons presented above regarding its independent claim (claim 35).

Claim 53:

Claim 53 is patentable for at least the reasons presented above regarding its independent claim (claim 45).

VIII. CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 1-45 was erroneous, and reversal of his decision is respectfully requested.

The Commissioner is authorized to charge the appeal brief fee of $500.00 and any other fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit

Account No. 501505/5181-61100/RCK. This Appeal Brief is submitted with a return receipt postcard.

Respectfully submitted,

Robert C. Kowert
Reg. No. 39,255
Attorney for Appellants

Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
(512) 853-8850

Date: March 9, 2005

IX. CLAIMS APPENDIX

The claims on appeal are as follows.

1. A method for configuring pluggable components, the method comprising:

configuring preference values for one or more pluggable components on a first device; and

distributing the one or more pluggable components to one or more other devices via a network subsequent to said configuring;

wherein the one or more pluggable components are executable within the one or more other devices in accordance with the configured preference values to provide services to users of the one or more other devices.

2. The method as recited in claim 1, wherein said configuring preference values for one or more pluggable components on a first device comprises:

receiving user input to a graphical user interface of the first device; and

modifying the preference values of a first of the one or more pluggable components in accordance with the received user input.

3. The method as recited in claim 2, further comprising displaying on the graphical user interface a current value of each of the preference values of the first pluggable component, wherein the received user input changes one or more of the displayed current values.

4. The method as recited in claim 2, further comprising validating the received user input prior to said modifying the preference values.

5. The method as recited in claim 1, wherein said configuring preference values for one or more pluggable components on a first device comprises:

receiving user input to a command line interface of the first device; and

modifying the preference values of a first of the one or more pluggable components in accordance with the received user input.

6. The method as recited in claim 5, wherein the received user input specifies one or more of the preference values of the first pluggable component and a new value for each of the specified preference values.

7. The method as recited in claim 5, further comprising validating the received user input prior to said modifying the preference values.

8. The method as recited in claim 1, wherein said configuring preference values of one or more pluggable components on a first device comprises modifying one or more of the preference values of at least one of the one or more pluggable components.

9. The method as recited in claim 1, further comprising initializing each of the preference values of each of the one or more pluggable components to a default value for the preference value prior to said configuring.

10. The method as recited in claim 1, wherein the one or more pluggable components is a plurality of pluggable components, wherein each of the plurality of pluggable components are copies of a first pluggable component.

11. The method as recited in claim 10, wherein the one or more other devices is a plurality of devices, wherein said configuring preference values comprises modifying

the preference values for each of the plurality of pluggable components for execution within a corresponding one of the plurality of devices.

12. The method as recited in claim 11, wherein said distributing comprises sending each of the plurality of pluggable components to the corresponding one of the plurality of devices via the network.

13. The method as recited in claim 1, wherein said configuring preference values for one or more pluggable components on a first device comprises:

generating a batch file comprising one or more configuration entries for the one or more pluggable components, wherein each configuration entry includes:

information specifying one of the one or more pluggable components;

information specifying one of the preference values for the specified pluggable component; and

a new value for the specified preference value; and

executing the batch file on the first device;

wherein said executing the batch file comprises executing each of the one or more configuration entries in the batch file, wherein each of the one or more configuration entries sets the specified preference value for the specified pluggable component to the new value of the configuration entry when executed.

14. The method as recited in claim 13,

wherein the batch file further comprises one or more distribution entries, wherein each distribution entry includes:

information specifying one of the one or more pluggable components; and

information specifying one of the one or more other devices as a destination device for the specified pluggable component;

wherein said executing the batch file comprises executing each of the one or more distribution entries in the batch file; and

wherein said executing each of the one or more distribution entries comprises sending the specified pluggable component to the specified destination device via the network.

15. The method as recited in claim 1, wherein each of the one or more pluggable components comprises a preferences file comprising the preference values associated with the pluggable component.

16. The method as recited in claim 15, wherein the preferences files are Java programming language Properties files.

17. The method as recited in claim 1, wherein each of the one or more other devices comprise an embedded server, wherein the one or more pluggable components are executable within the embedded server of each of the one or more other devices.

18. The method as recited in claim 16, wherein the embedded servers include Java Embedded Servers.

19. The method as recited in claim 1, wherein the pluggable components are Java Archive (JAR) files.

20. The method as recited in claim 1, wherein the network is the Internet.

21. A system comprising:

a first device; and

a plurality of devices operable to couple to the first device via a network;

wherein the first device is configured to:

configure preference values for a plurality of pluggable components in accordance with user input; and

distribute the plurality of pluggable components to the plurality of devices via the network subsequent to said configuring and in response to user input; and

wherein the plurality of pluggable components are executable within the plurality of devices in accordance with the configured preference values to provide services to users of the plurality of devices.

22. The system as recited in claim 21, wherein the first device comprises a display component, wherein, in said configuring preference values for a plurality of pluggable components, the first device is further configured to:

display in a graphical user interface on the display component a current value of each of the preference values of a first of the plurality of pluggable components;

receive user input to the graphical user interface changing one or more of the displayed current values; and

modify the preference values of the first pluggable component in accordance with the received user input.

23. The system as recited in claim 21, wherein the first device further comprises a display component, wherein, in said configuring preference values for a plurality of pluggable components, the first device is further configured to:

receive user input to a command line interface on the display component of the device, wherein the user input specifies one or more of the preference values of the first pluggable component and a new value for each of the specified preference values; and

modify the preference values of a first of the plurality of pluggable components in accordance with the received user input.

24. The system as recited in claim 21, wherein the processor is further operable to initialize each of the preference values of each of the plurality of pluggable components to a default value for the preference value prior to said configuring, and wherein, in said configuring preference values of the plurality of pluggable components, the first device is further configured to modify one or more of the default preference values of at least one of the plurality of pluggable components.

25. The system as recited in claim 21, wherein each of the plurality of pluggable components are copies of a first pluggable component, wherein, in said configuring preference values, the first device is further configured to customize the preference values for each of the plurality of pluggable components for execution within a corresponding one of the plurality of devices.

26. The system as recited in claim 25, wherein, in said distributing, the first device is further configured to send each of the plurality of pluggable components to the corresponding one of the plurality of devices via the network.

27. The system as recited in claim 21, wherein, in said configuring preference values for a plurality of pluggable components, the first device is further configured to:

generate a batch file comprising one or more configuration entries for the plurality of pluggable components in response to user input, wherein each configuration entry includes:

information specifying one of the plurality of pluggable components;

information specifying one of the preference values for the specified pluggable component; and

a new value for the specified preference value; and

execute the batch file;

wherein, in said executing the batch file, the first device is further configured to execute each of the one or more configuration entries in the batch file, wherein each of the one or more configuration entries sets the specified preference value for the specified pluggable component to the new value of the configuration entry when executed.

28. The system as recited in claim 27,

wherein the batch file further comprises one or more distribution entries, wherein each distribution entry includes:

information specifying one of the plurality of pluggable components; and

information specifying one of the plurality of devices as a destination device for the specified pluggable component;

wherein, in said executing the batch file, the first device is further configured to execute each of the one or more distribution entries in the batch file; and

wherein, in said executing each of the one or more distribution entries, the first device is further configured to send the specified pluggable component to the specified destination device via the network.

29. The system as recited in claim 21, wherein each of the plurality of pluggable components comprises a preferences file comprising the preference values associated with the pluggable component.

30. The system as recited in claim 21, wherein the preferences files are Java programming language Properties files.

31. The system as recited in claim 21, wherein each of the plurality of devices comprise an embedded server, wherein the plurality of pluggable components are executable within the embedded servers of the plurality of devices.

32. The system as recited in claim 31, wherein the embedded servers include Java Embedded Servers.

33. The system as recited in claim 21, wherein the pluggable components are Java Archive (JAR) files.

34. The system as recited in claim 21, wherein the network is the Internet.

35. A device comprising:

a memory configured to store program instructions;

an input device configured to receive user input; and

a processor configured to read the program instructions from the memory and to execute the program instructions, wherein, in response to execution of the program instructions, the processor is operable to:

configure preference values for one or more pluggable components on the device in accordance with received user input; and

distribute the one or more pluggable components to one or more other devices via a network subsequent to said configuring and in response to user input;

wherein the one or more pluggable components are executable within the one or more other devices in accordance with the configured preference values to provide services to users of the one or more other devices.

36. The device as recited in claim 35, further comprising:

a display component;

wherein, in said configuring preference values for one or more pluggable components, the processor is further operable to:

display in a graphical user interface on the display component a current value of each of the preference values of a first of the one or more pluggable components;

receive user input to the graphical user interface changing one or more of the displayed current values; and

modify the preference values of the first pluggable component in accordance with the received user input.

37. The device as recited in claim 35, further comprising:

a display component;

wherein, in said configuring preference values for one or more pluggable components, the processor is further operable to:

receive user input to a command line interface on the display component of the device, wherein the received user input specifies one or more of the preference values of the first pluggable component and a new value for each of the specified preference values; and

modify the preference values of a first of the one or more pluggable components in accordance with the received user input.

38. The device as recited in claim 35, wherein the processor is further operable to initialize each of the preference values of each of the one or more pluggable components to a default value for the preference value prior to said configuring.

39. The device as recited in claim 35, wherein the one or more pluggable components is a plurality of pluggable components, wherein each of the plurality of

pluggable components are copies of a first pluggable component, wherein the one or more other devices is a plurality of devices, wherein, in said configuring preference values, the processor is further operable to modify the preference values for each of the plurality of pluggable components for execution within a corresponding one of the plurality of devices.

40. The device as recited in claim 39, wherein, in said distributing, the processor is further operable to send each of the plurality of pluggable components to the corresponding one of the plurality of devices via the network.

41. The device as recited in claim 35, wherein, in said configuring preference values for one or more pluggable components on a first device, the processor is further operable to:

generate a batch file comprising one or more configuration entries for the one or more pluggable components in response to user input, wherein each configuration entry includes:

information specifying one of the one or more pluggable components;

information specifying one of the preference values for the specified pluggable component; and

a new value for the specified preference value; and

execute the batch file;

wherein, in said executing the batch file, the processor is further operable to execute each of the one or more configuration entries in the batch file, wherein each of the one or more configuration entries sets the specified

preference value for the specified pluggable component to the new value of the configuration entry when executed.

42. The device as recited in claim 35, wherein each of the one or more pluggable components comprises a preferences file comprising the preference values associated with the pluggable component.

43. The device as recited in claim 35, wherein the pluggable components are Java Archive (JAR) files.

44. The device as recited in claim 35, wherein the network is the Internet.

45. A carrier medium comprising program instructions, wherein the program instructions are computer-executable to implement:

configuring preference values for one or more pluggable components; and

distributing the one or more pluggable components to one or more devices via a network subsequent to said configuring;

wherein the one or more pluggable components are executable within the one or more devices in accordance with the configured preference values to provide services to users of the one or more devices.

46. The carrier medium as recited in claim 45, wherein, in said configuring preference values for one or more pluggable components, the program instructions are further computer-executable to implement:

displaying on a graphical user interface a current value of each of the preference values of a first of the one or more pluggable components;

receiving user input to the graphical user interface, wherein the received user input changes one or more of the displayed current values; and

modifying the preference values of the first pluggable component in accordance with the received user input.

47. The carrier medium as recited in claim 45, wherein, in said configuring preference values for one or more pluggable components, the program instructions are further computer-executable to implement:

receiving user input to a command line interface, wherein the received user input specifies one or more of the preference values of a first of the one or more pluggable components and a new value for each of the specified preference values; and

modifying the preference values of the first pluggable component in accordance with the received user input.

48. The carrier medium as recited in claim 45, wherein the program instructions are further computer-executable to implement initializing each of the preference values of each of the one or more pluggable components to a default value for the preference value prior to said configuring.

49. The carrier medium as recited in claim 45, wherein the one or more pluggable components is a plurality of pluggable components, wherein each of the plurality of pluggable components are copies of a first pluggable component, wherein the one or more devices is a plurality of devices, wherein, in said configuring preference values, the program instructions are further computer-executable to implement modifying the preference values for each of the plurality of pluggable components for execution within a corresponding one of the plurality of devices.

50. The carrier medium as recited in claim 49, wherein, in said distributing, the program instructions are further computer-executable to implement sending each of the plurality of pluggable components to the corresponding one of the plurality of devices via the network.

51. The carrier medium as recited in claim 45, wherein, in said configuring preference values for one or more pluggable components, the program instructions are further computer-executable to implement:

generating a batch file comprising one or more configuration entries for the one or more pluggable components, wherein each configuration entry includes:

information specifying one of the one or more pluggable components;

information specifying one of the preference values for the specified pluggable component; and

a new value for the specified preference value; and

executing each of the one or more configuration entries in the batch file, wherein each of the one or more configuration entries sets the specified preference value for the specified pluggable component to the new value of the configuration entry when executed.

52. The carrier medium as recited in claim 45, wherein each of the one or more devices comprise an embedded server, wherein the one or more pluggable components are executable within the embedded server of each of the one or more devices.

53. The carrier medium as recited in claim 52, wherein the pluggable components are Java Archive (JAR) files.

54. The carrier medium as recited in claim 45, wherein the network is the Internet.

X. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

XI. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.